Understanding the Scalability of an Unstructured CFD Simulation

Dinesh Kaushik

Computer Science Dept., Old Dominion University

Mathematics & Computer Science Div., ANL

David Keyes

Mathematics & Statistics Dept., Old Dominion University

Inst. for Scientific Computing Research, LLNL

Inst. for Computer Applics. in Sci. and Eng., NASA Langley

Workshop issues addressed

- What are characteristics of applications that scale well?
- What are indicators of bad scaling?
- What "tricks" exist to achieve better scaling?
- What is the status of parallel math libraries?
- What are the implications of processors-in-memory (PIM)?

from the perspective of a PDE-based workload

Plan of presentation

- "Enlightening" case history: PETSc-FUN3D
  - steady unstructured grid CFD simulation
  - highly tuned for per-processor performance
  - scaled to thousand of nodes (ASCI experiences)
  - Gordon Bell "special" category winner, 1999
  - characteristic of grid-based PDE simulations (e.g., RT, MHD)
- Discussion of solver software toolkit
  - freely available software "engine" behind case history
  - undergoing further DOE-sponsored development
  - novel solver algorithm for next scaling "jump" to 100,000 processors

Principal reference

- *High Performance Parallel Implicit CFD*, Gropp, Kaushik, Keyes & Smith, 2001, Parallel Computing 27, pp. 337-362
- Ancient processors (0.25-0.6 GHz), but reasonably contemporary discussion of scalability issues
- More balanced look at performance than this talk: per-node performance, algorithmic (convergence) scaling, hardware scaling
- Available on-line at

  www.math.odu.edu/~keyes/papers.html

Motivation

- **No computer system is well balanced for *all* computational tasks, or even for all phases of a *single* well-defined task, like solving nonlinear systems arising from discretizations of fluid dynamical systems.**
- **Given the need for high performance in the solution of these and related systems, one should be aware of which computational phases are limited by which aspect of hardware or software.**
- **With this knowledge, one can design algorithms to "play to" the strengths of a machine of given architecture, or one can intelligently select or evolve architectures for preferred algorithms.**

Features of app: PETSc-FUN3D

- Based on “legacy” (but contemporary) NASA CFD application, with significant F77 code reuse
- Portable, message-passing library-based parallelization, runs on NT boxes through Tflop/s ASCI platforms
- Simple multithreaded extension (for SMP Clusters)
- Sparse, unstructured data, implying memory indirection with only modest reuse
- Wide applicability to other implicitly discretized multiple-scale PDE workloads — of interagency, interdisciplinary interest

“Standard” implementation strategy

- Follow the “owner computes” rule under the dual constraints of minimizing the number of messages and overlapping communication with computation
- Each processor “ghosts” its stencil dependences in its neighbors
- Ghost nodes ordered after contiguous owned nodes
- Domain mapped from (user) global ordering into local orderings
- Scatter/gather operations created between local sequential vectors and global distributed vectors, based on runtime connectivity patterns
- Newton-Krylov-Schwarz implicit solver operations translated into SPMD local tasks and communication tasks
- Profiling used to help eliminate performance bugs in communication and memory hierarchy

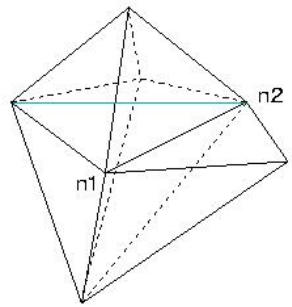
Primary PDE solution kernels*

- **Vertex-based loops**
  - state vector and auxiliary vector updates
- **Edge-based "stencil op" loops**
  - residual evaluation
  - approximate Jacobian evaluation
  - Jacobian-vector product (often replaced with matrix-free form, involving residual evaluation)
  - intergrid transfer (coarse/fine) in multilevel methods
- **Sparse, narrow-band recurrences**
  - approximate factorization and back substitution
  - smoothing
- **Vector inner products and norms**
  - orthogonalization/conjugation
  - convergence progress and stability checks

*assumes vertex-based; dual statements for cell-based

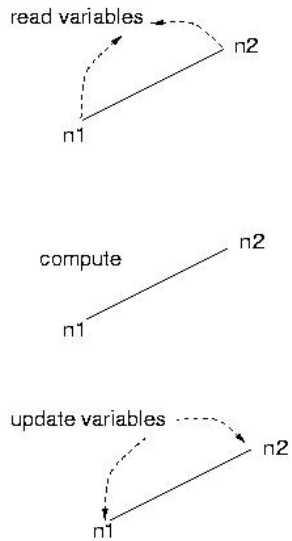
Illustration of edge-based loop

- Vertex-centered grid
- Traverse by edges
  - load vertex values
  - compute intensively
    - e.g., for compressible flows, solve 5x5 eigen-problem for character-istic directions and speeds of each wave
  - store flux contributions at vertices
- Each vertex appears in approximately 15 flux computations



Variables at each node:
density,
momentum (*x,y,z*),
energy,
pressure

Variables at edge:
identity of nodes,
orientation(*x,y,z*)
normal area



Complexities of PDE kernels

- **Vertex-based loops**
  - work and data closely proportional
  - pointwise concurrency, no communication
- **Edge-based "stencil op" loops**
  - large ratio of work to data
  - colored edge concurrency; local communication
- **Sparse, narrow-band recurrences**
  - work and data closely proportional
  - "frontal" concurrency; no, local, or global communication, depending on ordering and edges removed (algorithmic decision)
- **Vector inner products and norms**
  - work and data closely proportional
  - pointwise concurrency; global synchronization

Potential architectural stresspoints of PDE kernels

- **Vertex-based loops:**
  - memory bandwidth
- **Edge-based "stencil op" loops:**
  - load/store (register-cache) bandwidth
  - internode bandwidth
- **Sparse, narrow-band recurrences:**
  - memory bandwidth
  - internode bandwidth, internode latency, network diameter
- **Inner products and norms:**
  - memory bandwidth
  - internode latency, network diameter
- **ALL STEPS:**
  - memory latency, unless good locality is consciously built-in

Four potential limiters on arithmetic performance within a processor

- **Memory latency**
  - failure to predict which data items are needed
- **Memory bandwidth**
  - failure to deliver data at consumption rate of processor
- **Load/store instruction issue rate**
  - failure of processor to issue enough loads/stores per cycle, relative to surplus functional units and cached data
- **Floating point instruction issue rate**
  - low percentage of floating point operations among all operations (particularly in unstructured code)

Four potential limiters on scalability for multiple processors

- **Insufficient localized concurrency**
  - **poor solver algorithm (all other phases are pointwise concurrent)**
- **Load imbalance at synchronization points**
  - **poor mesh partitioner**
- **Interprocessor message latency**
  - **too many messages of small size**
- **Interprocessor message bandwidth**
  - **too high a communication volume**

Observation #1:
Processor scalability no problem*

- As popularized with the 1986 Karp Prize paper of Benner, Gustafson & Montry, Amdahl's law can be defeated if serial (or bounded concurrency) sections make up a decreasing fraction of total work as problem size and processor count scale – true for most iterative implicit nonlinear PDE solvers
- Simple, back-of-envelope parallel complexity analyses show that processors can be increased as fast, or almost as fast, as problem size, assuming load is perfectly balanced
- Caveat: the processor network must also be scalable (applies to protocols as well as to hardware); machines based on common bus networks will not scale

3D Stencil Costs (per Iteration)

- grid points in each direction ***n***, total work $N=O(n^3)$
- processors in each direction ***p***, total procs $P=O(p^3)$
- memory per node requirements $O(N/P)$
- execution time per iteration $A\, n^3/p^3$
- grid points on side of each processor subdomain ***n/p***
- neighbor communication per iteration $B\, n^2/p^2$
- cost of global reductions in each iteration $C \log p$ or $C\, p^{(1/d)}$
  - ***C*** includes synchronization frequency
- same dimensionless units for measuring ***A***, ***B***, ***C***
  - e.g., cost of scalar floating point multiply-add

3D Stencil Computation Illustration

Rich local network, tree-based global reductions

- **total wall-clock time per iteration**

$$T(n,p) = A\frac{n^3}{p^3} + B\frac{n^2}{p^2} + C\log p$$

- **for optimal *p*,** $\frac{\partial T}{\partial p} = 0$**, or** $-3A\frac{n^3}{p^4} - 2B\frac{n^2}{p^3} + \frac{C}{p} = 0,$

or (with $\theta \equiv \frac{32B^3}{243A^2C}$**),**

$$p_{opt} = \left(\frac{3A}{2C}\right)^{1/3}\left(\left[1+(1-\sqrt{\theta})\right]^{1/3} + \left[1-(1-\sqrt{\theta})\right]^{1/3}\right)\cdot n$$

- **without "speeddown," *p* can grow with *n***
- **in the limit as** $B/C \to 0$

$$p_{opt} = \left(\frac{3A}{C}\right)^{1/3}\cdot n$$

3D Stencil Computation Illustration

Rich local network, tree-based global reductions

- **optimal running time**

$$T(n, p_{opt}(n)) = \frac{A}{\rho^3} + \frac{B}{\rho^2} + C\log(\rho n),$$

where

$$\rho = \left(\frac{3A}{2C}\right)^{1/3}\left(\left[1+(1-\sqrt{\theta})\right]^{1/3} + \left[1-(1-\sqrt{\theta})\right]^{1/3}\right)$$

- **limit of infinite neighbor bandwidth, zero neighbor latency ($B \to 0$)**

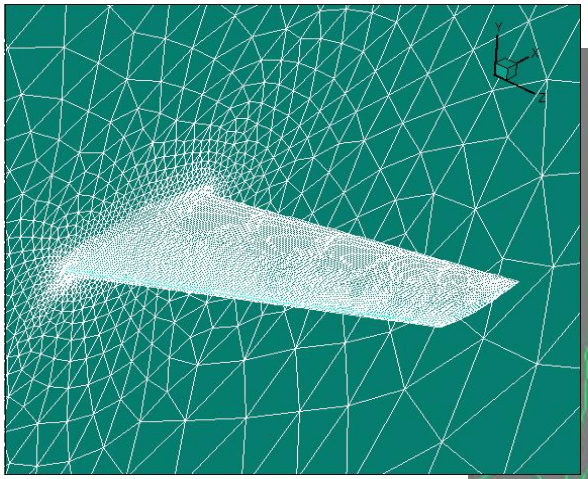
$$T(n, p_{opt}(n)) = C\left[\log n + \frac{1}{3}\log\frac{A}{C} + const.\right]$$

(This analysis is on a per iteration basis; fuller analysis would multiply this cost by an iteration count estimate that generally depends on *n* and *p*.)

Scalability of domain-decomposed bulk-synchronized PDE stencil computations

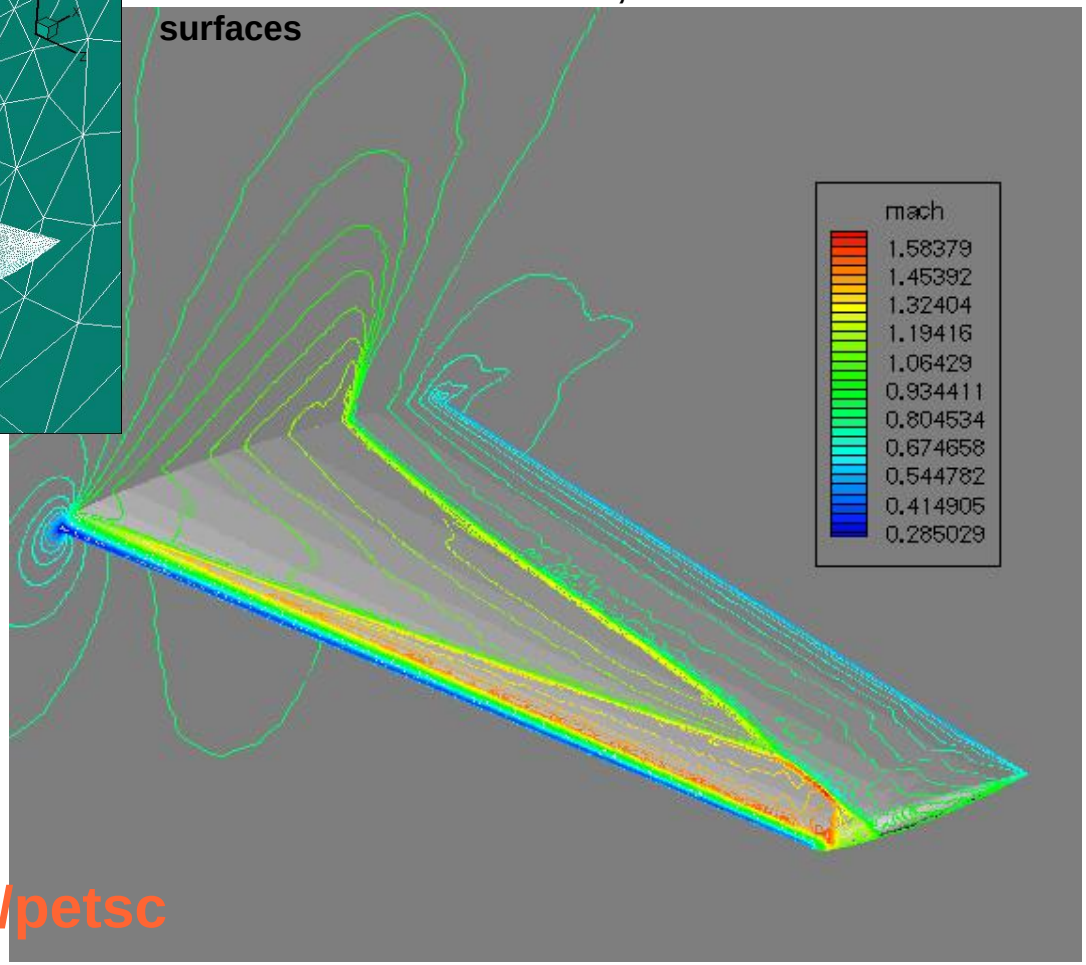
- **With tree-based (logarithmic) global reductions and scalable nearest neighbor hardware:**
  - **optimal number of processors scales *linearly* with problem size**
- **With 3D torus-based global reductions and scalable nearest neighbor hardware:**
  - **optimal number of processors scales as *three-fourths* power of problem size (almost "scalable")**
- **With common bus (point of neighbor pair contention):**
  - **optimal number of processors scales as *one-fourth* power of problem size (not "scalable")**

Computational aerodynamics



mesh c/o D. Mavriplis, ICASE

Transonic "Lambda" Shock, Mach contours on surfaces

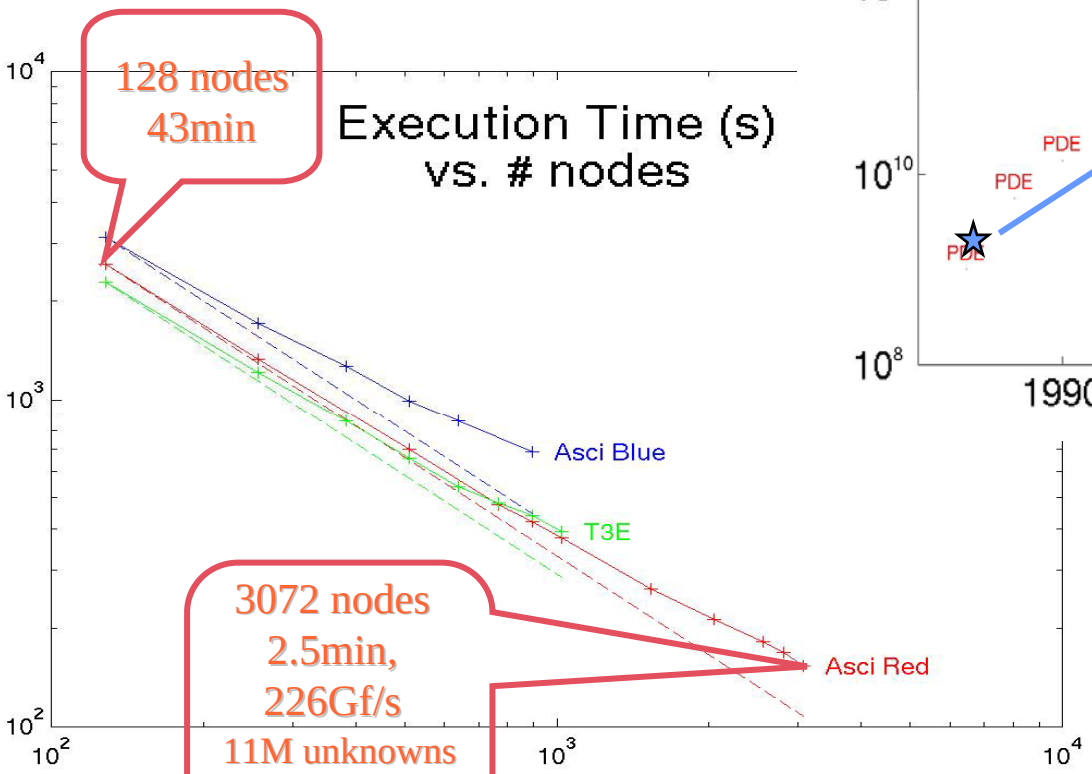


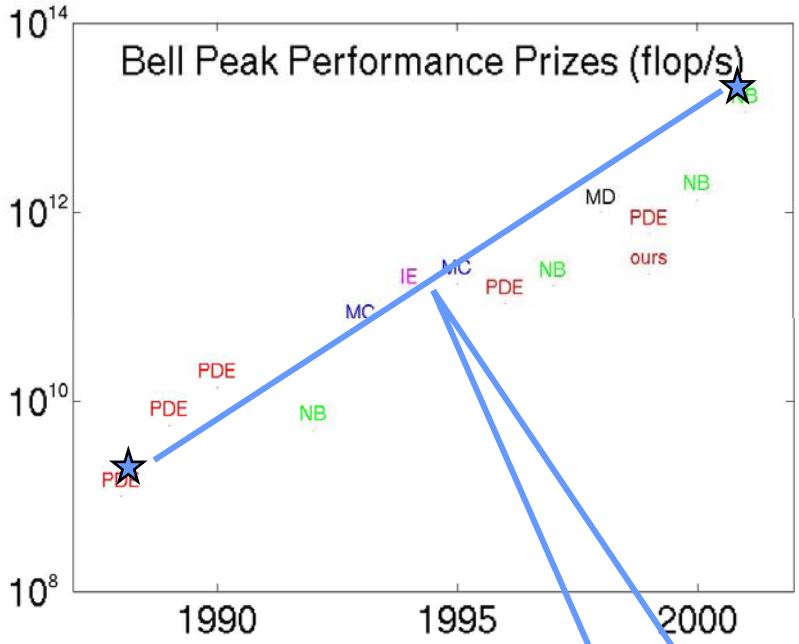
Implemented in PETSc

www.mcs.anl.gov/petsc

Fixed-size parallel scaling for NKS

This scaling study, featuring our widest range of processor number, was done for the ***incompressible*** case.

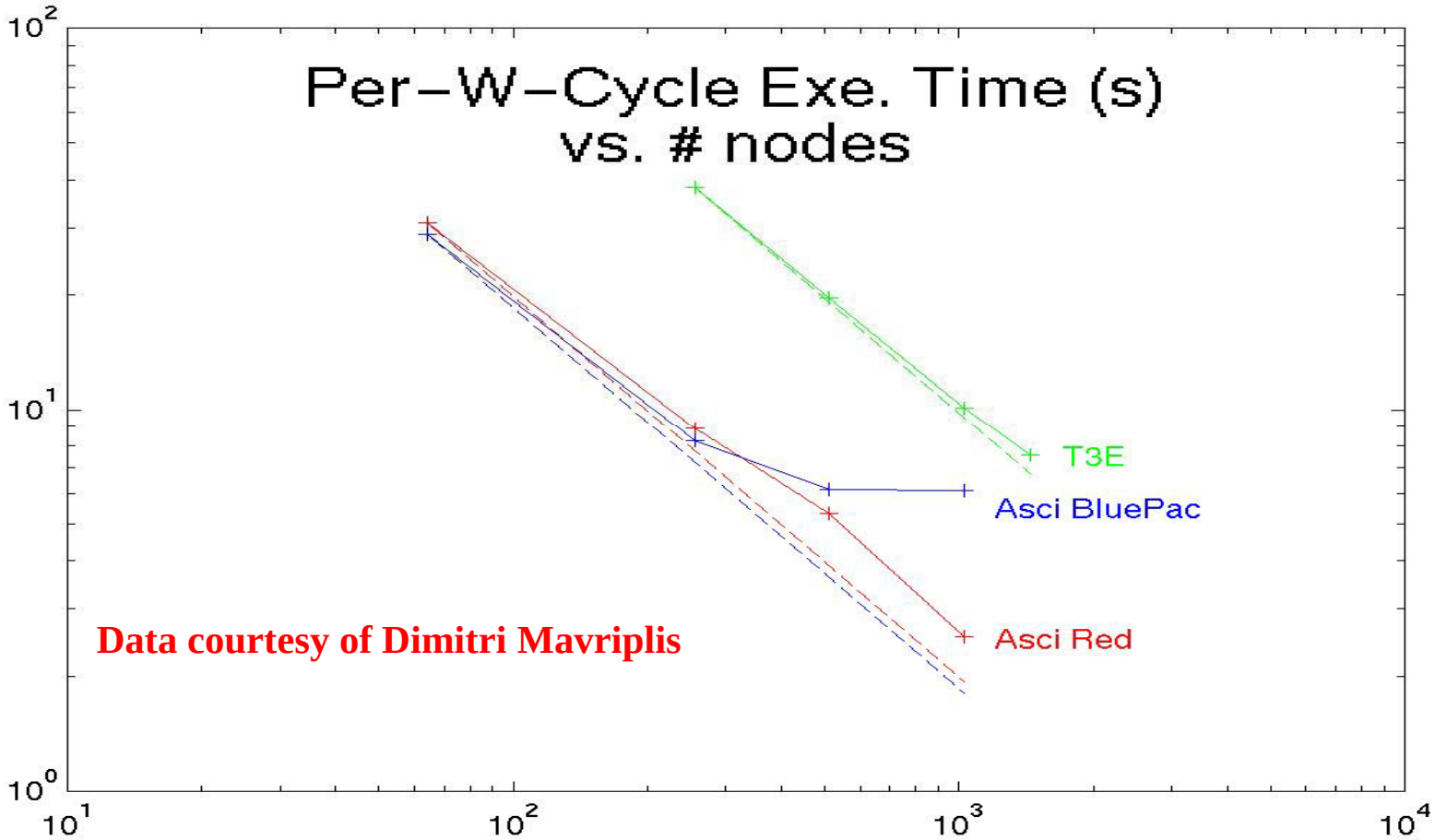




Four orders of magnitude in 13 years

c/o K. Anderson, W. Gropp, D. Kaushik, D. Keyes and B. Smith

Fixed-size parallel scaling for W-cycle MG



ASCI runs: for grid of 3.1M vertices; T3E runs: for grid of 24.7M vertices

Observation #2:
Synchronization eventually a bottleneck

- Percentage of time spent in communication phases on ASCI Red for NKS unstructured Euler simulation
- Principal nonscaling feature is synchronization at global inner products and norms, while cost of halo exchange grows slowly even for fixed-size problem with deteriorating surface-to-volume

Number of Processors	Global reductions	Synchronizations	Halo Exchanges
128	5%	4%	3%
256	3%	6%	4%
512	3%	7%	5%
768	3%	8%	5%
1024	3%	10%	6%

Observation #3:
Memory latency no problem*

- Regularity of reference in static grid-based computations can be exploited through memory-assist features to cover latency
- PDEs have simple, periodic workingset structure that permits effective use of prefetch/dispatch directives, and lots of slackness (process concurrency in excess of hardware concurrency)
- Combined with coming processors-in-memory (PIM) technology for gather/scatter into densely used block transfers and multithreading for latency that cannot be amortized by sufficiently large block transfers, the solution of PDEs can approach zero stall conditions
- Caveat: high bandwidth is critical to covering latency

* in principle

Improvements from locality reordering

Processor	Clock MHz	Peak Mflop/s	Opt. % of Peak	Opt. Mflop/s	Reord. Only Mflop/s	Intrl. only Mflop/s	Orig. Mflop/s	Orig. % of Peak
R10000	250	500	25.4	127	74	59	26	5.2
P3	200	800	20.3	163	87	68	32	4.0
P2SC (2 card)	120	480	21.4	101	51	35	13	2.7
P2SC (4 card)	120	480	24.3	117	59	40	15	3.1
604e	332	664	9.9	66	43	31	15	2.3
Alpha 21164	450	900	8.3	75	39	32	14	1.6
Alpha 21164	600	1200	7.6	91	47	37	16	1.3
Ultra II	300	600	12.5	75	42	35	18	3.0
Ultra II	360	720	13.0	94	54	47	25	3.5
Ultra II/HPC	400	800	8.9	71	47	36	20	2.5
Pent. II/LIN	400	400	20.8	83	52	47	33	8.3
Pent. II/NT	400	400	19.5	78	49	49	31	7.8
Pent. Pro	200	200	21.0	42	27	26	16	8.0
Pent. Pro	333	333	18.8	60	40	36	21	6.3

Factor of Five!

Data layouts - reorderings

- **Edge Reordering**
  - **sort the nodes at the ends of the edges**
  - **effectively transforms an edge based loop into a node based loop**
  - **enhances temporal locality**
- **Node Reordering**
  - **bandwidth reducing orderings will reduce the TLB and cache misses by referring to data items that are close in memory**
  - **our experience is with Reverse Cuthill-McGee and Sloan**

Enhancing locality

- **Choose data layouts that enhance locality at every level of memory hierarchy**
  - **Blocks for Registers**
    - **Block storage format for multicomponent systems – cuts the number of register loads**
  - **Interlaced Data Structures for Cache**
    - **Choose**

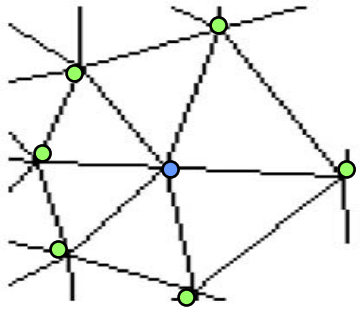
      u1,v1,w1,p1,u2,v2,w2,p2,…

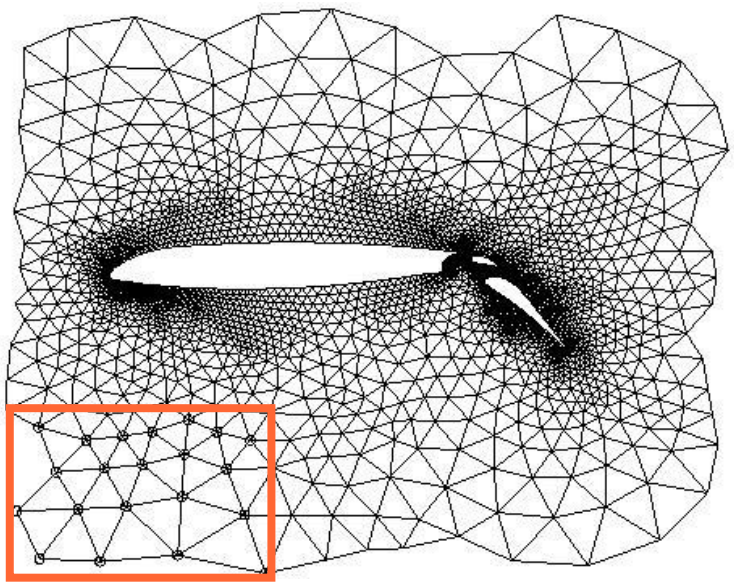
      in place of

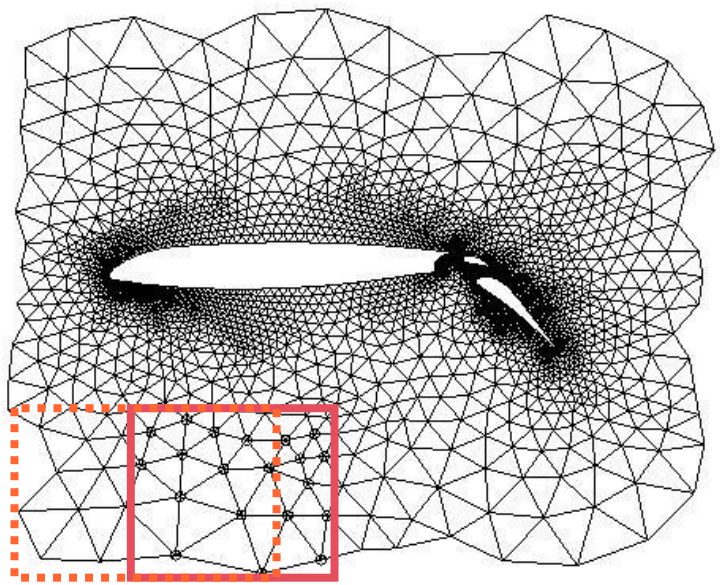
      u1,u2,…,v1,v2,…,w1,w2,…,p1,p2,…
    - **Cuts down TLB and data cache misses**
  - **Subdomains for Distributed Memory**
    - **“Chunky” domain decomposition for optimal surface-to-volume (communication-to-computation) ratio**

PDE workingsets

- Smallest: data for single stencil
- Largest: data for entire subdomain
- Intermediate: data for a neighborhood collection of stencils, reused as possible

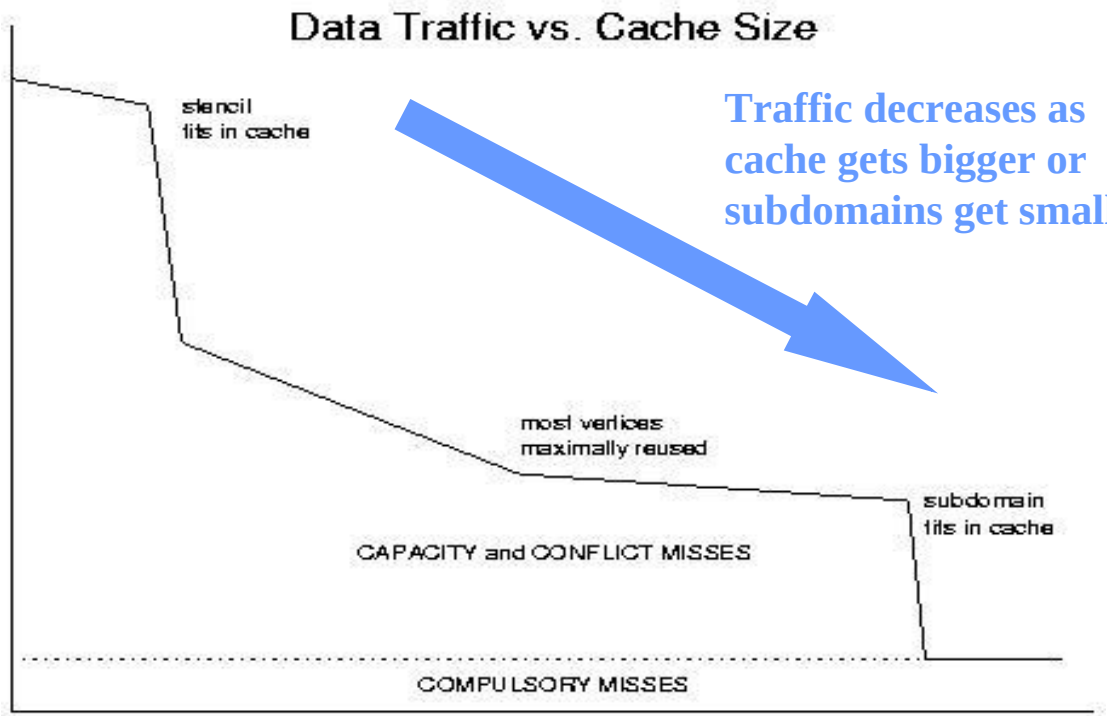




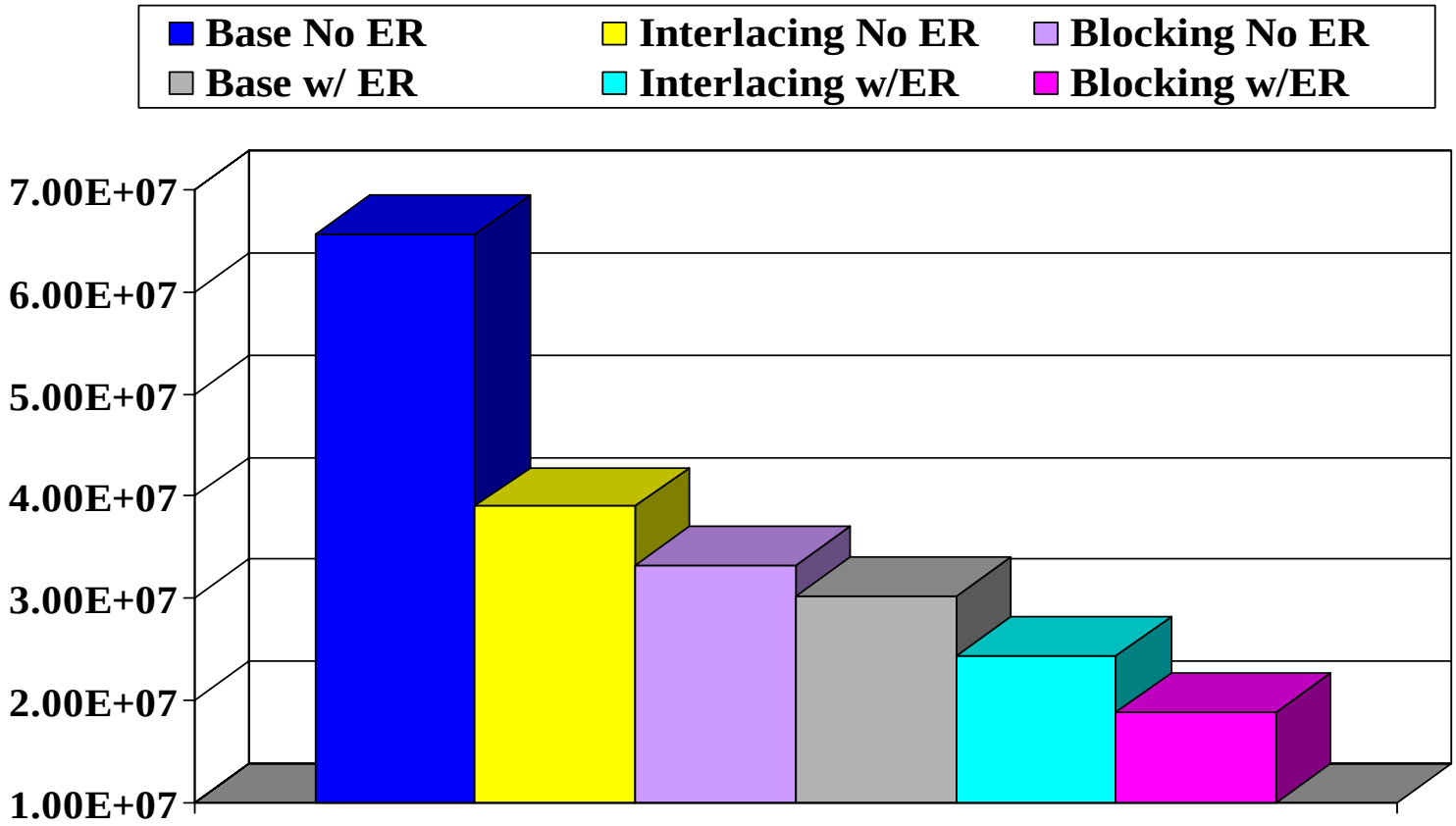


Cache Traffic for PDEs

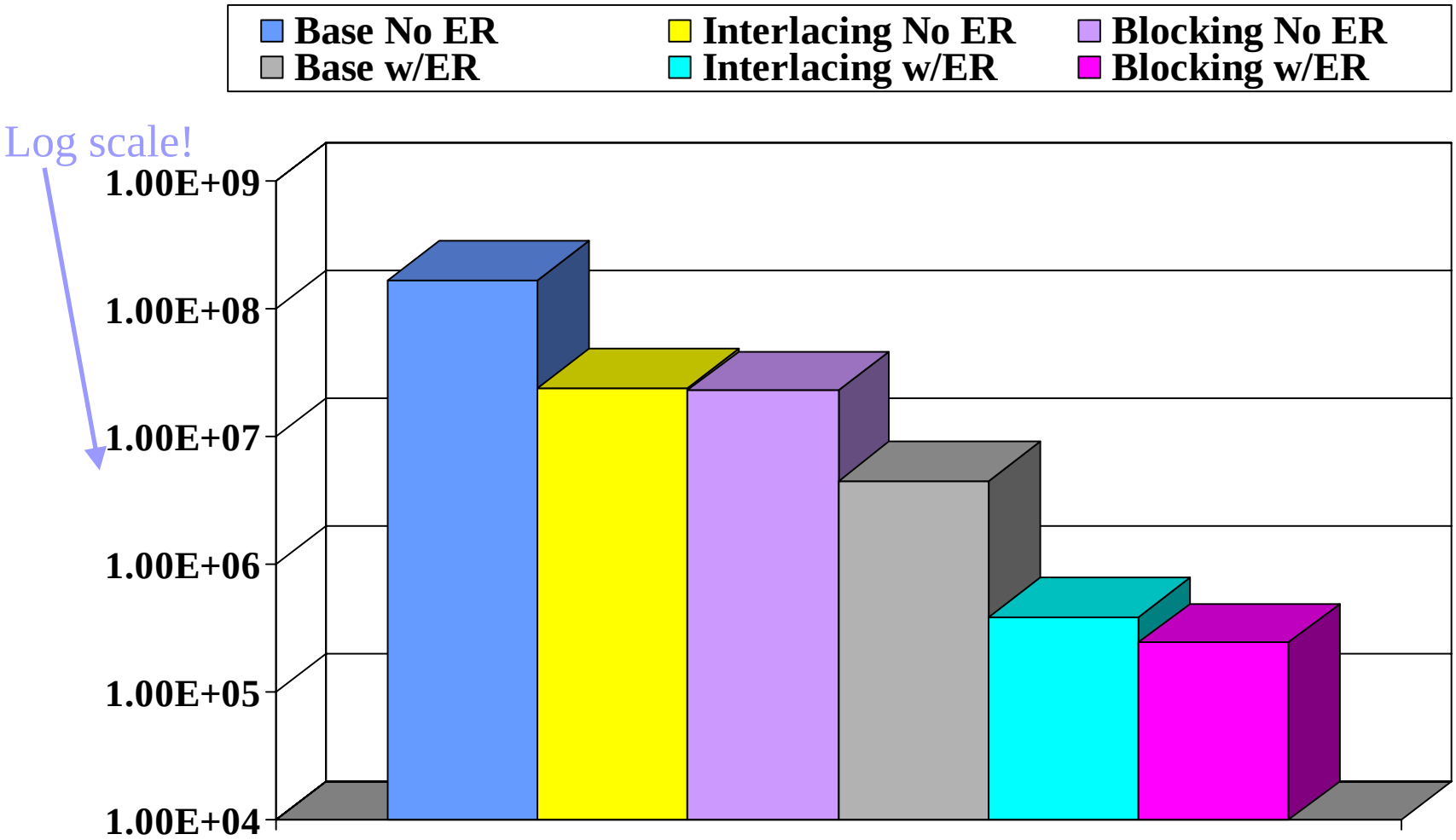
- **As successive workingsets "drop" into a level of memory, capacity (and with effort conflict) misses disappear, leaving only compulsory, reducing demand on main memory bandwidth**



Secondary cache misses: measured values on Origin



TLB misses: measured values on Origin



Observation #4
Memory bandwidth a major bottleneck

Execution times for NKS Euler Simulation on Origin 2000: (standard) double precision matrices versus single precision

Number of Processors	Computational Phase			
	Linear Solve		Overall	
	Double	Single	Double	Single
16	223s	136s	746s	657s
32	117s	67s	373s	331s
64	60s	34s	205s	181s
120	31s	16s	122s	106s

Note that times are nearly halved, along with precision, for the BW-limited linear solve phase, indicating that the BW can be at least doubled before hitting the next bottleneck!

ASCI memory bandwidth bottleneck

- **Per-processor memory bandwidth versus rate of work**
  - approximately 10-15 flops per word transferred from memory
  - fairly constant across machines, and fairly poor without extensive reuse

	Peak (MF/s)	BW/proc (MW/s)	(MF/s)/ (MW/s)
White	1500	125.0	12.0
Blue Mtn	500	48.8	10.2
Blue Pac	666	45.0	14.8
Red	333	33.3	10.0

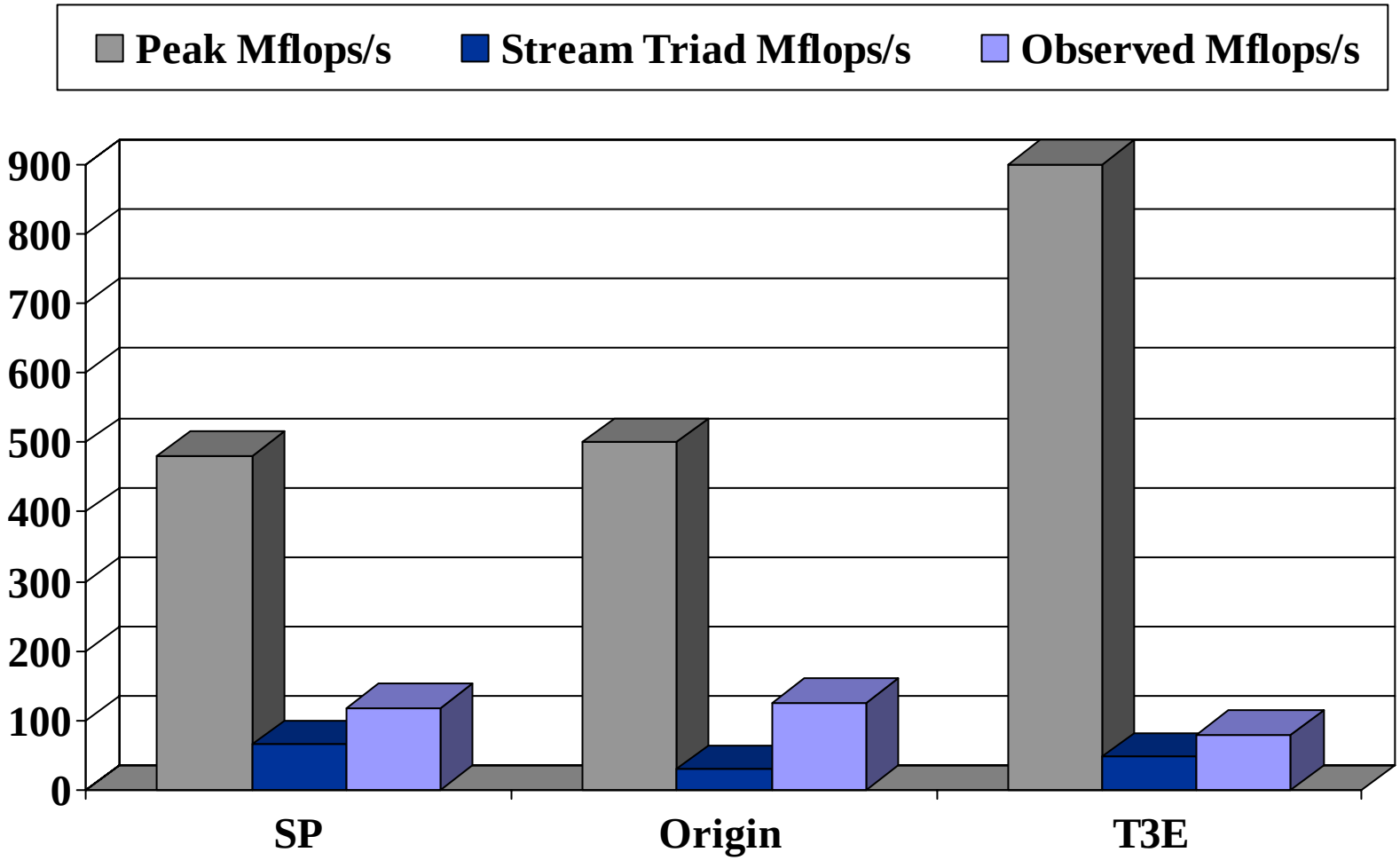
Implications of bandwidth limitations in shared memory systems

- The processors on a node compete for the available memory bandwidth
- The computational phases that are memory bandwidth limited will not scale and may even run more slowly due to arbitration
- Stream Benchmark on ASCI Red MB/s for the Triad Operation

Vector Size	1 Thread	2 Threads
1e04	666	1296
5e04	137	238
1e05	140	144
1e06	145	141
1e07	157	152

Larger vectors in last three rows do not fit into cache and are bandwidth-limited

PETSc-FUN3D sequential performance



Observation #5:
Load-store units may be bottleneck

- Table shows execution times of *residual flux evaluation phase* for NKS Euler simulation on ASCI Red (2 processors per node)
- Under either threads or message-passing, the second processor per node contributes another load/store unit while sharing fixed memory bandwidth
- Note that 1 thread is worse than 1 MPI process, but that 2-thread performance eventually surpass 2-process performance as subdomains become small

Nodes	MPI/OpenMP		MPI	
	1 Thr	2 Thr	1 Proc	2 Proc
256	483s	261s	456s	258s
2560	76s	39s	72s	45s
3072	66s	33s	62s	40s

Observation #6:
Fraction of flops may be bottleneck

```
do i=1, m
        jrow = ia(i+1)                                   // 1Of, AT, Ld
        ncol =  ia(i+1) -ia(i)                           // 1 Iop
        Initialize, sum_1 .....sum_N                     //  N Ld
        do j=1,ncol                                      // 1 Ld
          fetch ja(jrow), a(jrow), x_1(ja(jrow)), .....x_N(ja(jrow))
                                                         // 1 Of, N+2 AT N+2 Ld
          do N fmadd (floating multiply add)             // 2N Flop
        enddo                                            // 1 Iop, 1 Br
        Store sum_1.....sum_N in y_1(i) .....y_N(i)      // 1 Of, N AT, and St
enddo                                                    // 1 Iop, 1 Br
```

AT:address transln; **Br**: branch; **Iop**: integer op; **Flop**: floating point op; **Of**: offset calculation; **Ld**: load; **St**: store

- **Estimated number of floating point operations out of the total instructions ("intensity") for the unstructured Euler Jacobian**
  - **For** $N=1$, $I_f = 0.18$, **less than *one-fifth* of "peak" performance**
  - **For** $N = 4$, $I_f = 0.34$; **this is *one-third* of "peak" performance**

Summary of observations for PDE codes

- **Processor scalability no problem, in principle, provided there is a sufficiently rich interconnection network (mesh/torus okay for PDEs)**
- **Synchronization is eventually a bottleneck**
- **Memory latency no problem, in principle, with proper locality-based ordering**
- **Memory bandwidth is a *major* bottleneck**
- **Load-Store functionality *may* be a bottleneck in some physics kernels on some imbalanced processors**
- **Low frequency of floating point instructions is an algorithmic bottleneck intrinsic to unstructured problems**

Implications of PIM

- **The good news**
  - **With integer ops, PIM can very usefully support locality-based orderings (*à la* "gather-scatter" on vector machines) to improve spatial and temporal locality of use of the elements of *large dense vector gridfunctions* and *large sparse Jacobian matrices arising from local conservation laws***
  - **With floating ops, PIM can very usefully reduce memory traffic for simple kernels (e.g., vector triads and reductions) that otherwise severely tax memory bandwidth**
- **The bad news**
  - **PIM cannot solve the memory bandwidth problem completely**

More implications of PIM

- **While locality gains can be obtained "by hand rolling" today, PIM support is important for *dynamic adaptive gridding* versions**
  - **Results quoted herein are for static grids**
  - **SciDAC and general scientific computing thrusts require that this work be *redone* dynamically as grids are adapted**
  - **As illustrated, can lead to 2.5-fold to 7-fold improvements in per processor performance on unstructured PDE codes**
- **PIM also has potential to improve message-passing performance by off-loading buffer copies from the processor and its traffic from the CPU-memory system pipe**

Algorithm: Newton-Krylov-Schwarz

(See Cai, Gropp, Keyes & Tidriri (1994))

Newton

nonlinear solver

asymptotically quadratic

Krylov

accelerator

spectrally adaptive

Schwarz

preconditioner

parallelizable

Additive Schwarz Preconditioned Inexact Newton (ASPIN)

- **Nonlinear Schwarz moves most of the Newton work *inside* (local) just a few outer (global) iterations**
- **It replaces $F(u)=0$ with a new nonlinear system possessing the same root, $\Phi(u)=0$**
- **Define a correction $\delta_i(u)$ to the i^{th} partition (e.g., subdomain) of the solution vector by solving the following local nonlinear system:**

$$R_i F(u+\delta_i(u))=0$$

where $\delta_i(u)\in\Re^n$ is nonzero only in the components of the i^{th} partition

- **Then sum the corrections: $\Phi(u)=\sum_i \delta_i(u)$**

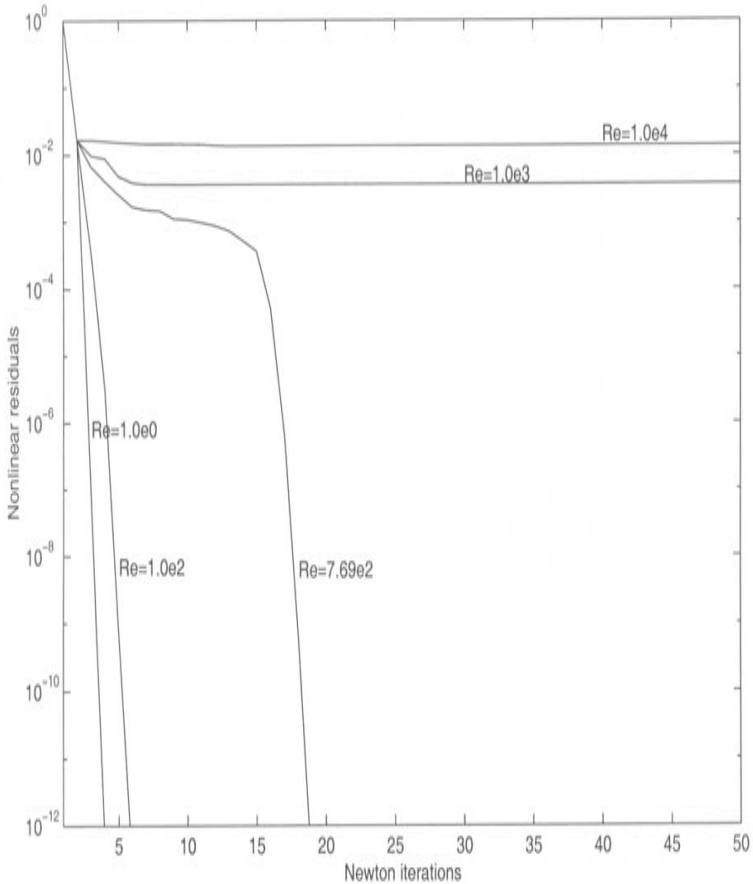
ASPIN, cont.

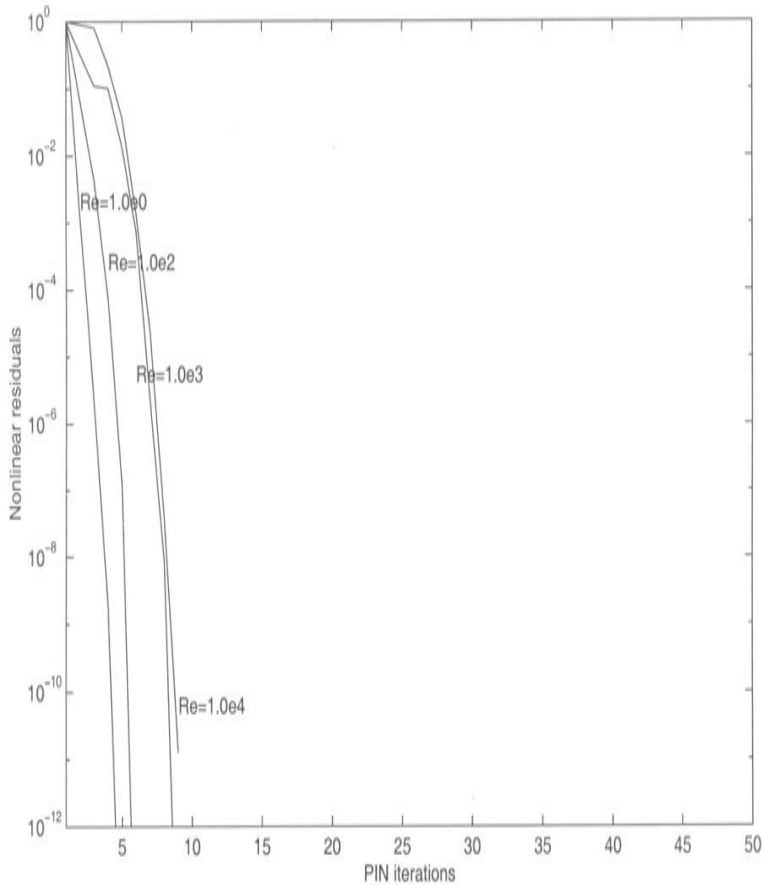
- It is simple to prove that if the Jacobian of $F(u)$ is nonsingular in a neighborhood of the desired root then $\Phi(u) = 0$ and $F(u) = 0$ have the same unique root
- To lead to a Jacobian-free Newton-Krylov algorithm we need to be able to evaluate for any $u, v \in \Re^n$:
  - the residual $\Phi(u) = \sum_i \delta_i(u)$
  - the Jacobian-vector product $\Phi(u)'v$
- Remarkably, (Cai-Keyes, 2002) it can be shown that all required operator actions are available in terms the original function $F(u)$

Architectural virtues of ASPIN

- Instead of frequent global synchronizations to carry out the Newton-Krylov iteration on a global Newton update vector, most of the progress towards the root is made in local subclusters, with synchronization spanning the subcluster
  - e.g., one in Pittsburgh, one in Illinois, one in San Diego
- The outer Newton-Krylov iteration executes relatively fewer times
- Load balance issues are still important

Robustification by-product of ASPIN





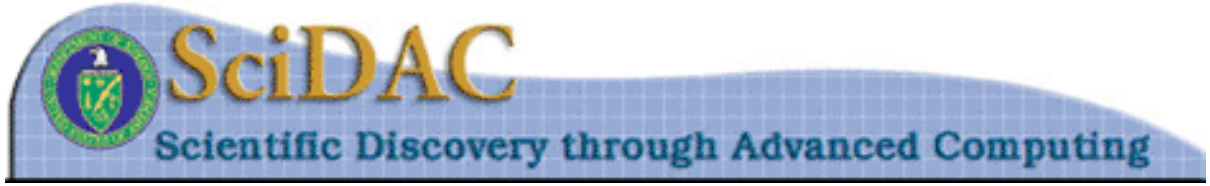
- **Driven cavity: Newton's method (left) versus new Additive Schwarz Preconditioned Inexact Newton (ASPIN) nonlinear preconditioning (right)**

Sources of higher performance

- **Algorithms that deliver more "science per flop"**
  - **possibly large problem-dependent factor, through adaptivity (but we won't count this towards rate improvement)**
- **Algorithmic variants that are more architecture-friendly**
  - **expect *half* an order of magnitude, through improved locality and relaxed synchronization**
- **More efficient use of processor cycles, and faster processor/memory**
  - **expect *one-and-a-half* orders of magnitude, through memory-assist language features, PIM, and multithreading**
- **Expanded number of processors**
  - **expect *two* orders of magnitude, through dynamic balancing and extreme care in implementation**

Take-home lessons for high-end CFD (or PDE) simulation

- **Unstructured (static) grid codes can run well on distributed hierarchical memory machines, with attention to partitioning, vertex ordering, component ordering, blocking, and tuning.**
- **Parallel scalability is easy, but attaining high per-processor performance for sparse problems gets more challenging with each machine generation.**
- ***Some* gains from hybrid parallel programming models (message passing and multithreading together) require little work; squeezing the last drop is likely much more difficult.**



- **Lab/university and physics/math/CS collaborations to advance computational applications to new levels of fidelity and create reusable, interoperable high-performance software**
- **Beginning FY2002, 51 new projects at $57M/year**
  - **Approximately one-third for "Integrated Software Infrastructure Centers" (ISICs)**
  - **A third for grid infrastructure and collaboratories**
  - **A third for applications groups**
- **5 Tflop/s IBM platform "Seaborg" at NERSC (#3 in Top 500) avail. for SciDAC contractors, plus new 5 Tflop/s "Cheetah" (IBM) at ORNL**

Terascale Optimal PDE Simulations

Who we are…

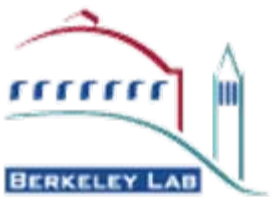
ARGONNE NATIONAL LABORATORY

Argonne National Laboratory is operated by The University of Chicago for the U.S. Department of Energy

… the PETSc and TAO people

Lawrence Livermore National Laboratory

… the hypre and PVODE people



Berkeley Lab

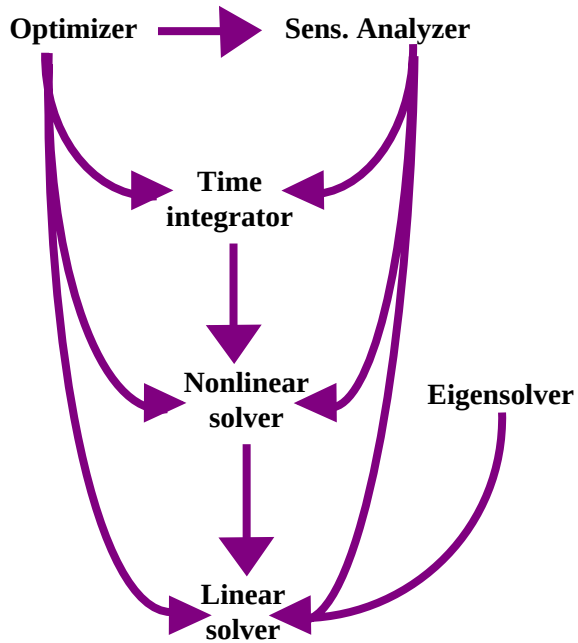
… the SuperLU and PARPACK people

Plus some university collaborators

Carnegie Mellon

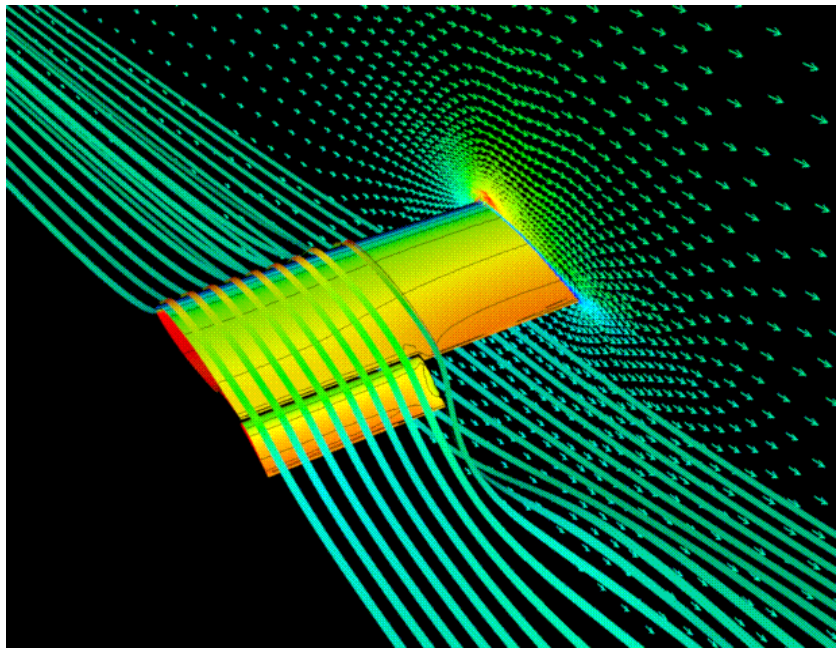
Scope for TOPS

- **Design and implementation of "solvers"**
  - **Time integrators (w/ sens. anal.)** $f(\dot{x}, x, t, p) = 0$
  - **Nonlinear solvers (w/ sens. anal.)** $F(x, p) = 0$
  - **Optimizers** $\min_u \phi(x,u)\ s.t.\ F(x,u) = 0, u \geq 0$
  - **Linear solvers** $Ax = b$
  - **Eigensolvers** $Ax = \lambda Bx$
- **Software integration**
- **Performance optimization**

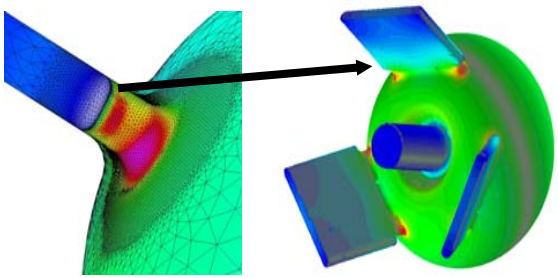


Salient application properties

- **Multirate**
  - **requiring fully or semi-implicit in time solvers**
- **Multiscale**
  - **requiring finest mesh spacing much smaller than domain diameter**
- **Multicomponent**
  - **requiring physics-informed preconditioners, transfer operators, and smoothers**



FUN3D Slotted Wing model, c/o Anderson et al.



PEP-II cavity model, c/o Advanced Computing for 21st Century Accelerator Science & Technology SciDAC group

- ***Not* linear, const. coefficient**
- ***Not* selfadjoint**
- ***Not* structured**

Motivation for TOPS

- Not just algorithms, but vertically integrated software suites
- Portable, scalable, extensible, tunable implementations
- Motivated by representative apps, intended for many others
- Featuring hypre and PETSc, among other existing packages
- Driven by three applications SciDAC groups
  - LBNL-led "21st Century Accelerator" designs
  - ORNL-led core collapse supernovae simulations
  - PPPL-led magnetic fusion energy simulations
- Coordinated with other ISIC SciDAC groups

- Many DOE mission-critical systems are modeled by PDEs
- Finite-dimensional models for infinite-dimensional PDEs must be large for accuracy
  - Qualitative insight is not enough (Hamming notwithstanding)
  - Simulations must resolve policy controversies, in some cases
- Algorithms are as important as hardware in supporting simulation
  - Easily demonstrated for PDEs in the period 1945–2000
  - Continuous problems provide exploitable hierarchy of approximation models, creating hope for "optimal" algorithms
- Software lags both hardware and algorithms

Workshop issues addressed

- What are characteristics of applications that scale well?
- What are indicators of bad scaling?
- What “tricks” exist to achieve better scaling?
- What is the status of parallel math libraries?
- What are the implications of processors-in-memory (PIM)?

Acknowledgments

- **Collaborators:**
  - **Kyle Anderson (NASA)**
  - **Xiao-Chuan Cai (CU-Boulder)**
  - **Dimitri Mavriplis (ICASE)**
  - **and the PETSc team at Argonne National Laboratory:**

    Satish Balay, Bill Gropp, Lois McInnes, Barry Smith
- **Sponsors: DOE, ICASE, NASA, NSF**
- **Computer Resources: DOE, SGI-Cray**

Bibliography

- *Globalized Newton-Krylov-Schwarz Algorithms and Software for Parallel CFD*, Gropp, Keyes, McInnes & Tidriri, 2000, Int. J. High Performance Computing Applications 14:102-136.
- *Four Horizons for Enhancing the Performance of Parallel Simulations based on Partial Differential Equations*, Keyes, 2000, Lecture Notes in Computer Science 1900:1-17.
- *Trends in Algorithms for Nonuniform Applications on Hierarchical Distributed Architectures*, Keyes, 2000, in *Computational Aerosciences for the 21st Century* (M. D. Salas and W. K. Anderson, eds.), Kluwer, Dordrecht, pp. 103-137.
- *High Performance Parallel Implicit CFD*, Gropp, Kaushik, Keyes & Smith, 2001, Parallel Computing 27, pp. 337-362.
- *Nonlinearly Preconditioned Inexact Newton Algorithms*, Cai & Keyes, 2002, *SIAM J. Sci. Comp.* (to appear)

All click-downloadable at …

Related URLs

- **Homepage for follow-up (talks, papers, pointers)**

  http://www.math.odu.edu/~keyes

- **FUN3D CFD code**

  http://www.mcs.anl.gov/petsc-fun3d

- **PETSc software**

  http://www.mcs.anl.gov/petsc

- **TOPS SciDAC project**

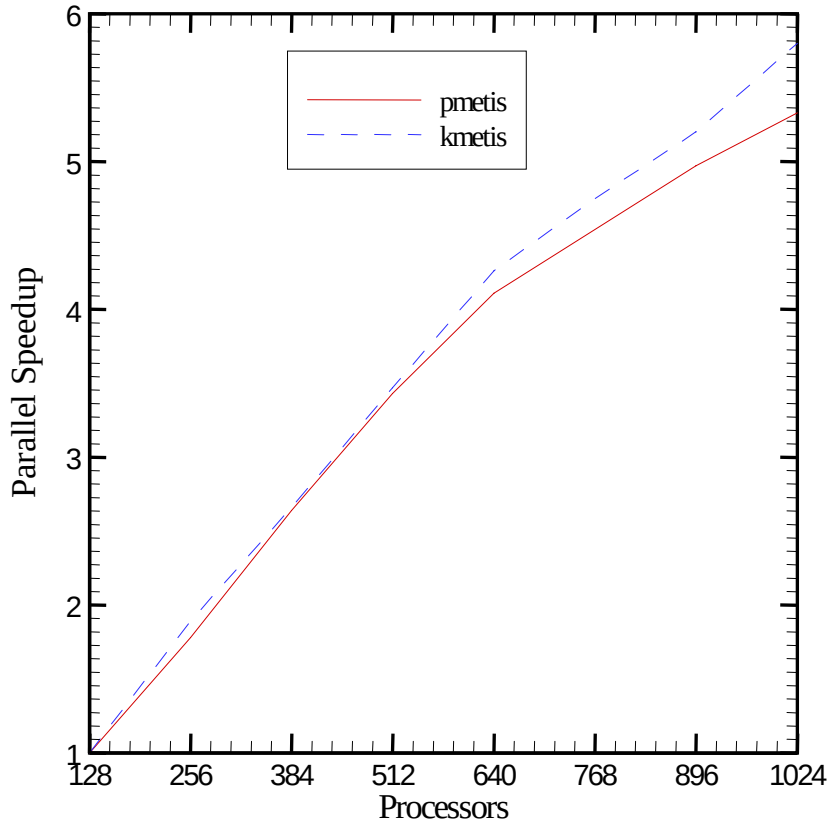
  http://www.math.odu.edu/~keyes/scidac

Extra Slides

Effect of Data Partitioning Strategies

Grid of 2.8 million vertices on Cray T3E-1200

- ***pmetis*** **attempts to balance the number of nodes and edges on each partition**
- ***kmetis*** **tries to reduce the number of non-contiguous subdomains and connectivity of the subdomains**
- ***kmetis*** **gives slightly better scalability**



Effect of Overlap and Fill on Convergence (contd.)

Execution times (in seconds) and linear iterations on a 333 MHz Pentium Pro ASCI Red for 358K-vertex case, with **ILU(1)**

Number of Processors	Overlap					
	0		1		2	
	Time	Linear Iterations	Time	Linear Iterations	Time	Linear Iterations
32	598	674	564	549	617	532
64	334	746	335	617	359	551
128	177	807	178	630	200	555

Effect of Overlap and Fill on Convergence (contd.)

Execution times (in seconds) and linear iterations on a 333 MHz Pentium Pro ASCI Red for 358K-vertex case, with **ILU(0)**

Number of Processors	Overlap					
	0		1		2	
	Time	Linear Iterations	Time	Linear Iterations	Time	Linear Iterations
32	688	930	661	816	696	813
64	371	993	374	876	418	887
128	210	1052	230	988	222	872

Effect of Overlap and Fill on Convergence (contd.)

Execution times (in seconds) and linear iterations on a 333 MHz Pentium Pro ASCI Red for 358K-vertex case, with **ILU(2)**

Number of Processors	Overlap					
	0		1		2	
	Time	Linear Iterations	Time	Linear Iterations	Time	Linear Iterations
32	688	527	786	441	-	-
64	386	608	441	488	531	448
128	193	631	272	540	313	472